МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ  
ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №4 по курсу объектно-ориентированное программирование I семестр, 2021/22 уч. год

Студент *Борисов Ян Артурович, группа М80-208Б-20*

Преподаватель *Дорохов Евгений Павлович*

### Цель работы

Целью лабораторной работы является:

· Закрепление навыков работы с классами.

· Создание простых динамических структур данных.

· Работа с объектами, передаваемыми «по значению».

### Задание

Необходимо спроектировать и запрограммировать на языке C++ класс-контейнер первого уровня, содержащий **одну фигуру ( колонка фигура 1),** согласно вариантам задания. Классы должны удовлетворять следующим правилам:

Требования к классу фигуры аналогичны требованиям из лаб.работы 1.

Классы фигур должны содержать набор следующих методов:  
  
 Перегруженный оператор ввода координат вершин фигуры из потока std::istream (>>). Он должен заменить конструктор, принимающий координаты вершин из стандартного потока.

Перегруженный оператор вывода в поток std::ostream (<<), заменяющий метод Print из лабораторной работы 1.

Оператор копирования (=)

Оператор сравнения с такими же фигурами (==)

Класс-контейнер должен соджержать объекты фигур “по значению” (не по ссылке).

Класс-контейнер должен содержать набор следующих методов:

Нельзя использовать:

· Стандартные контейнеры std.

· Шаблоны (template).

· Различные варианты умных указателей (shared\_ptr, weak\_ptr).

Программа должна позволять:

· Вводить произвольное количество фигур и добавлять их в контейнер.

· Распечатывать содержимое контейнера.

· Удалять фигуры из контейнера.

**Дневник отладки**

Во время выполнения лабораторной работы программа была несколько раз отлажена, так как плохо работала функция удаления из дерева. После нескольких отладок программа стала работать исправно.

**Недочёты**  
Недочётов не было обнаружено.

**Выводы**

Лабораторная работа №4 - это модернизация последних лабораторных 2 семестра. Если на 1 курсе я реализовывал вектор при помощи структур на языке СИ, то сейчас я реализовал вектор при помощи ООП на языке С++. Лабораторная прошла успешно, я повторил старый материал и узнал, усвоил много нового.

**Исходный код**

Tvector.h

#pragma once

#include "square.h"

class TVector {

public:

TVector();

TVector(const TVector &vector);

virtual ~TVector();

void InsertLast(const Square &square);

Square RemoveLast();

const Square &Last();

Square &operator[](const size\_t index);

size\_t Length();

bool Empty();

void Clear();

friend std::ostream &operator<<(std::ostream &os, const TVector &vector);

private:

Square \*data\_;

size\_t size\_, capacity\_;

};

Tvector.cpp

#include "tvector.h"

#include <cstring>

enum {

CAPACITY = 32,

};

TVector::TVector()

: data\_(nullptr), size\_(0), capacity\_(CAPACITY)

{

data\_ = (Square \*)malloc(capacity\_ \* sizeof(Square));

}

TVector::TVector(const TVector &vector)

: data\_(nullptr), size\_(vector.size\_), capacity\_(vector.capacity\_)

{

data\_ = (Square \*)malloc(vector.capacity\_ \* sizeof(Square));

std::memcpy(data\_, vector.data\_, capacity\_ \* sizeof(Square));

}

TVector::~TVector()

{

free(data\_);

data\_ = nullptr;

size\_ = 0;

capacity\_ = 0;

}

size\_t TVector::Length()

{

return size\_;

}

bool TVector::Empty()

{

return !size\_;

}

Square &TVector::operator[](const size\_t index)

{

return data\_[index];

}

void TVector::InsertLast(const Square &square)

{

if (size\_ >= capacity\_) {

capacity\_ <<= 1;

data\_ = (Square \*)realloc(data\_, capacity\_ \* sizeof(Square));

}

data\_[size\_++] = square;

}

Square TVector::RemoveLast()

{

return data\_[--size\_];

}

const Square &TVector::Last()

{

return data\_[size\_ - 1];

}

void TVector::Clear()

{

size\_ = 0;

capacity\_ = CAPACITY;

data\_ = (Square \*)realloc(data\_, capacity\_ \* sizeof(Square));

}

std::ostream &operator<<(std::ostream &os, const TVector &vector)

{

for (size\_t i = 0; i < vector.size\_; ++i)

os << vector.data\_[i];

return os;

}

Square.h

#include "figure.h"

#include <iostream>

#include "point.h"

class Square : public Figure {

public:

Square();

Square(std::istream &is);

virtual ~Square();

void Print(std::ostream& os);

double Area();

size\_t VertexesNumber();

private:

Point a, b, c, d;

};

Square.cpp

#include "square.h"

Square::Square() : a(), b(), c(), d() {

std::cout << "Default square was created" << std::endl;

}

Square::Square(std::istream &is) {

std::cout << "Enter lower left coordinate" << std::endl;

std::cin >> a;

std::cout << "Enter upper left coordinate" << std::endl;

std::cin >> b;

std::cout << "Enter upper right coordinate" << std::endl;

std::cin >> c;

std::cout << "Enter lower right coordinate" << std::endl;

std::cin >> d;

std::cout << "Square was created" << std::endl;

}

Square::~Square() {

std::cout << "Square deleted" << std::endl;

}

void Square::Print(std::ostream& os) {

std::cout << "Square: " << a << " " << b << " " << c << " " << d << std::endl;

}

double Square::Area() {

double len\_a = a.dist(b);

double len\_b = b.dist(c);

return len\_a \* len\_b;

}

size\_t Square::VertexesNumber() {

return 4;

}